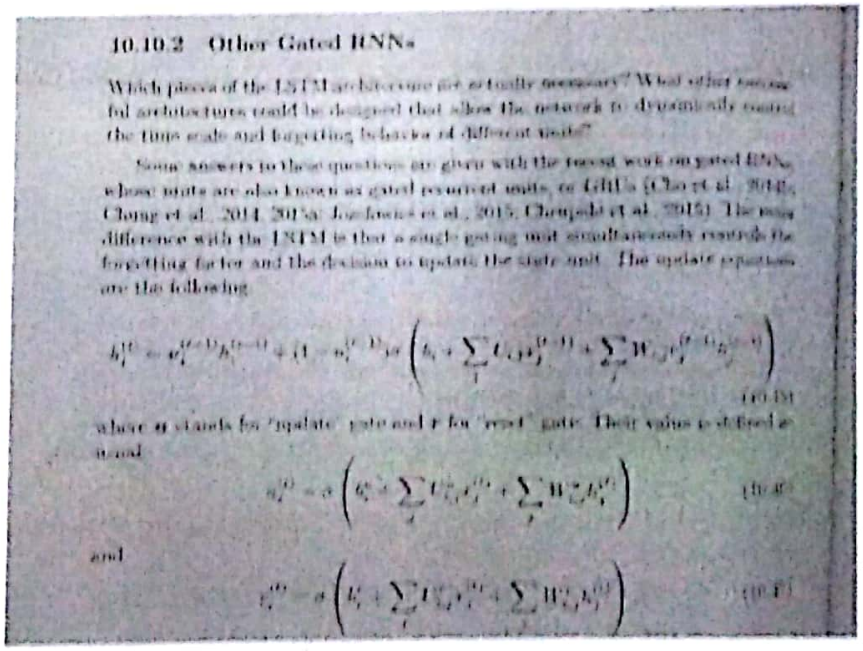
COL 865: Special Topics in Computer Applications - Deep Learning

Thursday October 5, 2017

Notes:

- **Time: 2:30 pm to 4:30 pm. Total Questions: 3. Maximum Points: 40.**
- **This question paper has printed material on both sides.**
- **This exam is open handwritten notes. Any kind of printed/typed material (e.g., photocopies of books, online notes) is not allowed.**
- **Some questions may be harder than others. Use your time wisely.**
- **For no-objective type answers, use a notation as close to the one used in class as possible.**
- **Start each answer on a new page.**
- **The answers to fill in the blank questions should be written on the question paper itself. You should attach the corresponding sheet with your answer sheet.**



10.10.2 Other Gated RNNs

Figure 1: Gated Recurrent Unit (GRU)

. **Sequence to Sequence Architectures:** In this problem, we will make use of bi-directional Gated Recurrent Units (GRUs) for a sequence to sequence learning task i.e., the input and the output sequences could be of different lengths.

- **(8 points)** Draw an encoder-decoder architecture where the input is processed using GRUs arranged in a bi-directional fashion. The output is generated using a sequence uni-directional GRUs. You should draw the rolled (compact) as well as the unrolled architecture.

- **(5 points)** Next, we will make this architecture deep. In particular, we would like to add another bi-directional GRU layer on top of the existing bi-directional GRU units while processing the input. Similarly, we would like to add another GRU layer (uni-directional) on top of the existing GRU layer while processing the output. How would your architecture change? As before, draw the rolled as well as the unrolled architectures.
- **(5 points)** Next, we would like to add attention to the network in the part above. Draw the new architecture with attention incorporated. In this case, you only need to draw the unrolled architecture.

For each of the parts above, you clearly need to specify the interconnections between different units and the associated parameters. State which parameters are tied with each other. Also, write down the equations describing input/output mappings between different units. You should make use of symbols/notations which are as close to the one covered in class as possible (for ease of evaluation). You can refer to Figure 1 as a reminder for the functionality of a GRU where the symbols are as described in class.

2. **Backpropagation.**

(a) **(5 points)** Derive the equations for backpropagation through a normalization unit in a multi-layered perceptron. You should make use of the following notation.

- $\mathcal{L}$: Loss function
- $x^{(k)}$: input to the k^{th} hidden layer
- $z^{(k)}$: result of linear transformation in the k^{th} layer
- $\hat{z}^{(k)}$: output of the normalizer placed at layer k.
- $g^{(k)}$: activation function in layer k

Specifically, you should derive the expression for $\nabla_{z^{(k)}}\mathcal{L}$ in terms of $\nabla_{\hat{z}^{(k)}}\mathcal{L}$, as well as the gradient $\nabla_{\theta_N}\mathcal{L}$ where θ_N is the set of parameters used by the normalization unit.

(b) Consider a CNN layer with input x sized $h \times w \times d$. Consider applying r kernels each of size $t \times t \times d$ on this input with a stride of 1. Let z denote the output tensor (i.e., 3d matrix) obtained after applying the convolution (linear) operation. Let z be of size $h' \times w' \times d'$. Let x' be the output obtained after applying a (pointwise) non-linear transformation using function g. Note that z and x' are of the same size. Assume there is no-zero padding.

- **(2 points)** Derive the expression for h', w' and d' in terms of h, w, d and t, r. You should briefly justify your answer.
- **(5 points)** Derive the expression for backpropagation through this convolutional layer. Specifically, you need to derive the expression for (1) $\frac{\partial \mathcal{L}}{\partial K_{a,b,l,l'}}$ and (2) $\frac{\partial \mathcal{L}}{\partial x_{i,j,l}}$ where $K_{a,b,l,l'}$ denotes the a, b, l^{th} entry of the l'^{th} kernel and $x_{i,j,l}$ denotes an entry in the 3d matrix x with index $i.j.l$. You can assume that you have the backpropagating gradients $\frac{\partial \mathcal{L}}{\partial x'_{i',j',l'}}$ available to you for this computation where $x'_{i',j',l'}$ denotes an entry in the 3d matrix x' with index i', j', l'. You can ignore the bias term for simplification.